Deepfake Video Detection via Facial Action Dependencies Estimation

Lingfeng Tan[1], **Yunhong Wang**[1], **Junfu Wang**[1], **Liang Yang**[2], **Xunxun Chen**[3], **Yuanfang Guo**[1 4 *]

[1] School of Computer Science and Engineering, Beihang University, China
[2] School of Artificial Intelligence, Hebei University of Technology, China
[3] CNCERT/CC, Beijing, China
[4] Zhongguancun Laboratory, Beijing, China
{tanlf, yhwang, junfuwang, andyguo}@buaa.edu.cn, yangliang@vip.qq.com, cxx@cert.org.cn

Abstract

Deepfake video detection has drawn significant attention from researchers due to the security issues induced by deepfake videos. Unfortunately, most of the existing deepfake detection approaches have not competently modeled the natural structures and movements of human faces. In this paper, we formulate the deepfake video detection problem into a graph classification task, and propose a novel paradigm named Facial Action Dependency Estimation (FADE) for deepfake video detection. We propose a Multi-Dependency Graph Module (MDGM) to capture abundant dependencies among facial action units, and extracts subtle clues in these dependencies. MDGM can be easily integrated into the existing frame-level detection schemes to provide significant performance gains. Extensive experiments demonstrate the superiority of our method against the state-of-the-art methods.

Introduction

With the advance in deep learning techniques, various deep neural network (DNN) based face manipulation techniques/softwares, such as FakeApp, ZAO, Deepfake (Deepfakes 2019), DeepFaceLab(Perov et al. 2020), have led to the proliferation of deepfake images/videos on the internet. These deepfake techniques can generate indistinguishable videos, and they may induce various security problems, including fake news, infringement of personal privacy, etc. Therefore, researches on deepfake image/video detection, especially the deepfake facial image/video detection, have become an important topic in the field of image/video forensics.

There exist two major types of deepfake video detection mechanisms, i.e., frame-level detection and video-level detection. Frame-level detection methods usually seek for subtle clues in a single frame, e.g. jitters, blurs, and strange artifacts (Zi et al. 2020). With extensive efforts devoted to facilitating the detection, schemes, such as exposing blending boundaries (Li et al. 2020a), detecting via frequency domain features (Qian et al. 2020; Liu et al. 2021), and leveraging multiple attentions on different regions (Zhao et al. 2021), etc., achieve superior performances. Unfortunately, the accuracy of these methods may not be quite satisfactory when serving for deepfake video detection, due to the ignorance of temporal dependencies.

In video-level detection methods, a portion of them focuses on modeling the temporal inconsistencies among the adjacent frames, including color jitters (Gu et al. 2021), offsets of facial landmarks (Sun et al. 2021), etc. Meanwhile, other detection methods model the bio-signatures, e.g., heart rate, eye-blinking (Ciftci, Demir, and Yin 2020a,b). These methods tend to assign more emphasis on the relationships between adjacent frames, while the intra-correlations within each single frame are utilized incompetently.

Although the existing methods show decent performances in deepfake video detection, they have not properly considered the facial movements and the structural properties of the human faces, which is a limitation of the existing methods. The natural facial actions are generally accomplished by several facial action units, which function together according to certain structural correlations. Meanwhile, sev-

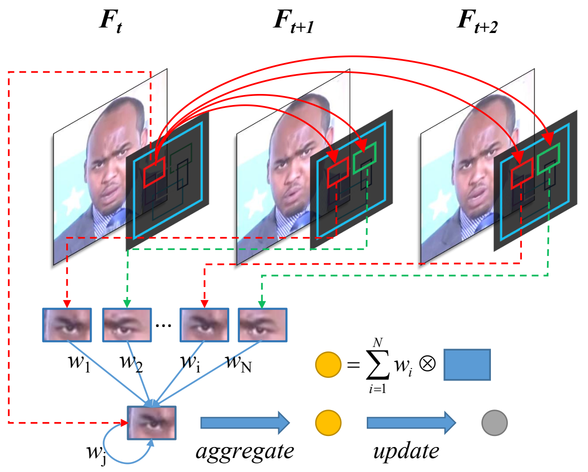

Figure 1: There exists various dependencies between the Facial Action Units on a face. The red edges represent the temporal action dependencies between the left eye and connected action units on a serious face. According to these dependencies, the left eye can exchange information with other action units and update its feature from a more comprehensive perspective.

*Corresponding author.

eral researches (Mittal et al. 2020; Hosler et al. 2021) have proven that existing manipulation methods cannot precisely simulate facial movements and express emotions.

Based on the above observations, we assume that the manipulation clues are more likely to be exposed in certain facial regions, where frequent movements happen, such as regions highly correlated with expressions. Besides, multiple dependencies among these regions across different frames (including long-distanced frames) tend to provide more comprehensive and robust clues, especially on real-like data and low-quality videos. Since the existing CNNs, RNNs and Transformers are not quite suitable for processing these local facial regions with complex dependencies across multiple frames (including long-distanced frames), we explicitly model these discrete regions with specific dependencies as a graph.

In this paper, we propose a novel deepfake video detection method by proposing a novel paradigm named Facial Action Dependency Estimation (FADE), to explore the subtle clues and dependencies in the facial action regions. Specifically, based on our assumption, we formulate the deepfake video detection problem as a graph classification task and leverage Graph Convolutional Networks (GCNs) to solve it. According to Facial Action Coding System (Ekman and Friesen 1978), the facial action units, which are highly correlated with facial movements, are selected as the nodes, and their features are extracted by a backbone network with the RoI Alignment (He et al. 2017). To construct the graph and perform graph classification, we propose a module named Multi-Dependency Graph Module (MDGM). In particular, MDGM contains four parts, i.e., Pattern-Mixer, Action Branch, Content Branch, and Fusion Block. The Pattern-Mixer can model the action dependencies in a more applicable manner for deepfake video detection. Then, the Action Branch can process the node features and the dependencies represented by Pattern Mixer, and generate a more robust feature. Meanwhile, the Content Branch aims to capture the dynamic dependencies according to the node features, as a complement to Action Branch. At last, we construct a Fusion Block, which combines and enhances the features extracted from different branches to generate a more comprehensive representation to achieve deepfake video detection.

Our major contributions are summarized as follows.

- We formulate the deepfake video detection as a graph classification task, according to structural prior information, and propose a novel Facial Action Dependency Estimation method for deepfake video detection.
- We propose a Multi-Dependency Graph Module, which can capture abundant dependencies among the facial action units, and extract clues in these dependencies. In addition, this module can be easily transplanted to the existing frame-level detection methods.
- We propose a Pattern Mixer, which can model the prior action dependencies from multiple perspectives and output the mixed patterns, to provide more appropriate dependencies for deepfake video detection.
- Extensive experiments have demonstrated the effectiveness of our proposed method against the state-of-the-art methods, and explain the underlying mechanisms of our methods via visualizations.

Related Work

Graph Convolutional Networks

Graph Convolutional Network (GCN) features good representation capability on structural data, which can be formulated as a graph. In the past few years, GCNs have been widely exploited to process large-scale structural data, such as biochemical compounds (Wale, Watson, and Karypis 2008), social networks (Hamilton, Ying, and Leskovec 2017), etc.

The excellent feature representation ability of GCNs have also been utilized in the field of computer vision. In video processing, such as video action recognition and video anomaly detection, GCNs play vital roles. By considering the structure of human skeletons, GCNs are adopted to model the movement of linked joints to recognize human actions (Jain et al. 2016; Yan, Xiong, and Lin 2018). In video anomaly detection, HL-Net (Wu et al. 2020) applies GCNs to capture the long-distance dependencies among the frames and multiple modalities. ACAD (Zhong et al. 2019) selects different correlations of frames, including features similarity and temporal consistency, to build the graphs, and achieves good performance.

Since GCNs can well model the correlations (especially the structural dependencies) in a group of features, we exploit GCNs to better capture the manipulation clues in the deepfake facial videos.

Deepfake Video Detection

Current deepfake video detection schemes can be classified into two categories, frame-level and video-level methods.

Frame-level methods prefer to excavate the inconsistencies between the manipulated and original regions in a single frame. Early frame-level methods usually rely on the hand-crafted or statistical features (Fridrich and Kodovsky 2012; Zhang, Zheng, and Thing 2017). Unfortunately, they are expired due to the evolving forgery techniques. With the rapid development of deep learning, CNN-based methods become dominant. GramNet (Liu, Qi, and Torr 2020) reveals the global texture differences between the real and fake images/frames. (Li et al. 2020a) proposes Face X-ray to detect the subtle clues of boundary-blending, which is an important step in faceswap. The anomaly artifacts in frequency domain, which are induced by the generation methods, are detected by F3-Net (Qian et al. 2020) and SPSL (Liu et al. 2021). By taking the personal identity into considerations (Cozzolino et al. 2021; Dong et al. 2022), unseen manipulations can be handled and the generalization capabilities can be improved. Unfortunately, these frame-level methods have not considered the temporal information due to their natures.

For video-level methods, they usually pay more attentions to the inconsistencies between frames, because most of the synthesis methods lack temporal constraints. (Li et al. 2020b) formulates the deepfake video detection as a multiple instance learning task to handle the partial face attack.

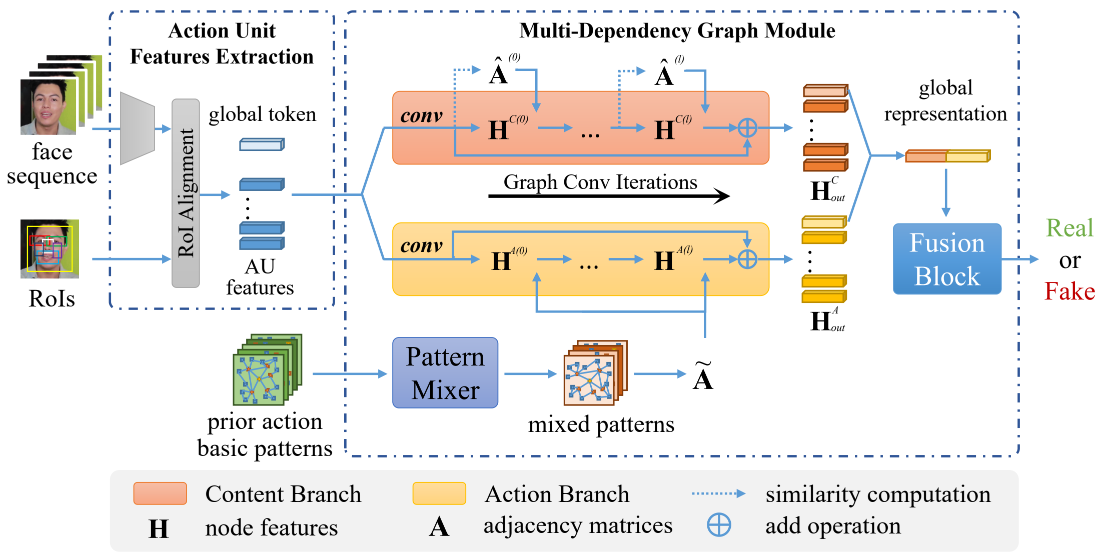

Figure 2: The architecture of our Facial Action Dependency Estimation for deepfake video detection.

Forensic methods based on bio-signatures also perform decently. Multiple approaches are developed to model the heart rate and identify the authenticity of videos (Ciftci, Demir, and Yin 2020b; Fernandes et al. 2019; Qi et al. 2020). LipForensics (Haliassos et al. 2021) considers the movement of lips in a video as a robust detection feature. (Gu et al. 2021) gradually amplifies the pixel jitters in the manipulated videos. These video-level methods tend to ignore the intra-correlations within each single frame.

PatchDiffusion (Zhang et al. 2022) directly crops the feature map into smaller patches, connects them via similarities to form the graph and straightforwardly applies a Graph Neural Network (GNN) to achieve a frame-level deepfake detection. On the contrary, our FADE formulates the deepfake video detection task as a graph classification task, constructs the graph according to the facial structural information as well as the facial action dependencies, generates comprehensive representations containing intra- and inter-frame information, and achieves video-level detection.

The Proposed Work

According to our above assumptions that the manipulation clues are more likely to be exposed in the facial regions related to facial expressions, and multiple dependencies can be explored to obtain more comprehensive representations, we can convert the input facial video into a graph. In the graph, the features of the facial action units are regarded as nodes, and the dependencies among them are the edges. Then, deepfake video detection is formulated as a graph classification task and we accordingly propose a novel deepfake video detection method, named Facial Action Dependency Estimation (FADE). FADE contains two stages: Action Unit Feature Extraction and Multi-Dependency Graph Module (MDGM). Fig. 2 illustrates the architecture of FADE.

Action Unit Feature Extraction

Facial Action Coding System (FACS) (Ekman and Friesen 1978) is a well-known theory which depicts the correlations between different facial muscle movements and different expressions. In FACS, Ekman and Friesen design fine-grained action units (AUs) based on the correlations and action characteristics of AUs. By considering the mechanisms of mainstream face manipulation approaches, the demand of deepfake detection and the limitations of face detection methods, we define six AUs in each face, including, the left eye, right eye, left cheek, right cheek, nose and mouth. To perform graph classification, we utilize the features of the entire face as the frame master node in each frame and a learnable global token as the global master node for the entire video. Under such circumstance, one frame master node is connected to all the AUs within its frame and the global master node is connected to every frame master node in the video. After message passing, the features in the global master node become the final representation of the video. We extract the AU features $\mathbf{X} \in \mathbb{R}^{N \times C \times h \times w}$ via a backbone with RoI Alignment (He et al. 2017) operation, where N, C, h, and w denote the number of nodes, the channel, the height, and the width of AU features, respectively.

Multi-Dependency Graph Module

The Multi-Dependency Graph Module (MDGM) is proposed to leverage GCNs to generate a robust feature for deepfake video detection, based on various dependencies and the node features. MDGM consists of a Pattern-Mixer, an Action Branch, a Content Branch, and a Fusion Block.

Pattern Mixer In expression classification task, the expressions are usually classified into 8 categories (Lucey et al. 2010), e.g., happy, sad, afraid, etc. The action dependencies and action unit activations usually vary for different expressions. However, there exists obvious overlaps in the moving AUs for different expressions. To better capture the action

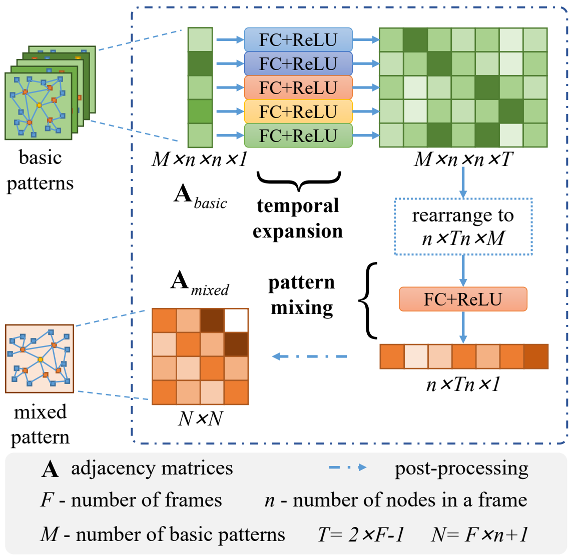

Figure 3: The procedures of Pattern Mixer. Multiple mixed patterns are generated simultaneously via identical procedures.

dependencies and avoid an overly fine-grained expression categorization for deepfake video detection, we empirically re-categorize the expressions into three classes, including neutral faces, smiling faces, and serious faces, instead of directly employing the original 8 expression categories. The new categorization can well cover all the common facial movements in typical deepfake videos. Three corresponding basic action patterns are then designed with reference to FACS. For example, there is only one connection between the left and right eyes on a neutral face. Meanwhile, the left eye, left cheek and mouth are usually correlated with each other on a smiling face, and similar correlations can be discovered on the right half of the face. Additional patterns are designed based on the positional relationships.

Since the hand-crafted dependencies may not be optimal for deepfake video detection. we propose a mixer-like method named Pattern Mixer, which can adjust the weights of edges and mix the basic patterns into more appropriate patterns, as shown in Fig. 3. Pattern Mixer consists of two steps, i.e., temporal expansion and pattern mixing. In temporal expansion, the basic patterns in a single frame are firstly assigned with different temporal weights according to the sequential order of the frames. Then, different patterns are mixed up to generate a new one, via linear mapping with activation functions. The input of pattern-mixer is the basic action patterns $\mathbf{A} \in \mathbb{R}^{M\times n\times n}$, where n indicates the number of AUs in a single frame. The output of the pattern-mixer is the mixed $\tilde{\mathbf{A}} \in \mathbb{R}^{\tilde{M}\times N\times N}$, where $\tilde{M}$ is set to 4, and N denotes the number of nodes in the entire video, including the global master node.

Action Branch The Action Branch aims to handle the graphs, which is constructed based on the mixed dependencies $\tilde{\mathbf{A}}$ from Pattern Mixer. The Action Branch consists of a projection layer and a ConvGCN. The projection layer is a convolutional layer with the kernel size of 1×1. The outputs of the projection layer are divided into four groups along the channel dimension.

In GCN, each layer is composed of an aggregation operation and an update operation. In the aggregation operation, nodes aggregate information from its neighbors, which can be formulated as

$$\mathbf{H}^{'} = \mathrm{Aggregate}(\mathbf{H},\mathbf{A}) = \mathbf{D}^{-0.5}\mathbf{A}\mathbf{D}^{-0.5}\mathbf{H}, \quad (1)$$

where $\mathbf{D}^{-0.5}\mathbf{A}\mathbf{D}^{-0.5}$ is the symmetric normalized adjacency matrix and $\mathbf{D}$ is the degree matrix determined by $\mathbf{A}$. $\mathbf{H}^{'} \in \mathbb{R}^{N\times C\times\ h\times w}$ is the collection of the aggregated node features which are the linear combinations of the local neighbors' features.

$\mathbf{H}$ in FADE is a feature map with the dimension of $N\times C\times\ h\times w$, rather than a vector in the standard GCN. Inspired by (Zhang, Wang, and Guo 2019), we modify the update process to retain more spatial information, and finally implement it as a **Conv-BN-ReLU** layer, i.e.,

$$\mathrm{Update}(\mathbf{H}^{'}) = \mathrm{ReLU}\left(\mathrm{BN}(\mathrm{Conv}_{3\times3}(\mathbf{H}^{'}))\right), \quad (2)$$

where $\mathrm{Conv}_{3\times3}(\cdot)$, $\mathrm{BN}(\cdot)$, and $\mathrm{ReLU}(\cdot)$ represent a convolutional layer with kernel size of 3×3, a batch normalization layer and a ReLU activation function, respectively.

Then, our graph convolution layer is formulated as

$$\mathbf{H}^{(l+1)} = \mathrm{Update}^{(l+1)}\left(\mathrm{Aggregate}(\mathbf{H}^{(l)},\mathbf{A})\right), \quad (3)$$

where the $\mathbf{H}^{(l+1)}$ and $\mathbf{H}^{(l)}$ denote the node features at the l-th layer and $(l+1)$-th layer, respectively, with $\mathbf{H}^{(0)} = \mathbf{X}$.

Inspired by (He et al. 2016), we add a residual connection in the network and the entire two-layered model is named ConvGCN. Then, the feature $\mathbf{H}_{out}$ can be finally obtained. The entire process is described as

$$\mathbf{H}_{out} = \mathrm{ConvGCN}(\mathbf{H}^{(0)},\mathbf{A}) = \mathbf{H}^{(2)} + \mathbf{H}^{(0)}, \quad (4)$$

where $\mathbf{H}^{A}_{out}$ is the output of the Action Branch.

Content Branch To mine more potential dependencies, we construct the Content Branch as a complement to the Action Branch. The architecture of Content Branch is identical to the Action Branch. However, the dependencies are dynamically estimated by calculating the similarities of node features in each iteration. We convert the feature map $\mathbf{H} \in \mathbb{R}^{N\times C\times h\times w}$ to a vector $\mathbf{h} \in \mathbb{R}^{N\times C}$ via max pooling, and calculate the cosine similarities. The process is described as

$$\mathbf{A}_{i,j} = \frac{h_i h_j^T}{(\|h_i\|\|h_j\| + \epsilon)\times\alpha}, \quad (5)$$

$$\hat{\mathbf{A}} = \mathbf{Softmax}(\mathbf{A}), \quad (6)$$

where ϵ and α stand for a scalar to prevent zero-division and a learnable parameter to amplify the dependency differences, respectively. The default value of ϵ is 0.01. After graph convolution in Content Branch, the features are denoted as $\mathbf{H}^{C}_{out}$.

Method	Param (M)	LQ ACC	LQ AUC	HQ ACC	HQ AUC
MesoNet	≤ 1.0	70.47	-	83.10	-
Xception(Avg)*	21.6	86.42	89.30	95.97	97.87
Xception(LSTM)*	55.2	86.06	90.32	95.63	98.41
F3-Net*	44.2	87.01	88.15	96.30	98.85
Face X-ray	≥ 43.0	-	61.60	-	87.40
Multi-Attention*	32.8	86.67	88.20	96.63	99.07
DIANet	27.8	89.77	94.50	96.37	98.80
PatchDiffusion	28.4	86.11	89.29	95.16	98.51
RECCE	24.7	91.03	95.02	97.06	99.32
Two-Branch	-	-	91.10	-	99.12
UV-Transformer	≥ 130	-	-	**99.52**	**99.64**
Finfer	-	-	-	-	95.67
FADE	25.6	**92.89**	**95.98**	98.33	99.52

Table 1: Performances on the FF++ dataset with HQ and LQ settings. Param stands for the (possible) number of parameters of these methods. Bold numbers represent the best result in the column and * denotes our implementations.

Method	cross-quality FF++(LQ)	cross-dataset CDF	cross-dataset DFD
MesoNet	-	54.80	96.30
Xception*	70.87	65.90	93.45
EfficientNet-b4*	69.55	64.29	94.22
DSP-FWA	62.00	59.93	90.14
Face X-ray	72.80	74.76	-
SPSL	-	76.88	-
PatchDiffusion	-	68.23	95.74
RECCE	-	68.71	-
Two-branch	-	73.41	-
LipForensics	-	**82.40**	-
Finfer	-	70.60	-
FADE	83.33	74.83	**98.41**
FADE†	**84.64**	77.46	96.23

Table 2: Cross-dataset and cross-quality evaluations on the Celeb-DF and DFD datasets while training on FF++. * represents our implementations and FADE†is constructed without the Content Branch.

Fusion and Classification The output features, $\mathbf{H}_{out}^{A}$ from the Action Branch and $\mathbf{H}_{out}^{C}$ from the Content Branch, are concatenated. Then, the concatenated features are fed to the Fusion Block, which contains two **Conv-BN-ReLU** layers with a kernel size 1×1. At last, the fused feature is sent to the classifier after a global average pooling.

$$\mathbf{H}_{fuse} = \text{Fusion}((\mathbf{H}_{\text{out}}^{\text{A}} \| \mathbf{H}_{\text{out}}^{\text{C}})), \tag{7}$$

$$\hat{p} = \text{Classifier}\left(\text{GAP}\left(\mathbf{H}_{\text{fuse}}\right)\right), \tag{8}$$

where $\text{GAP}(\cdot)$ and $\text{Classifier}(\cdot)$ represent the global average-pooling and the softmax layer, respectively. The Cross-Entropy Loss is employed as our classification loss.

Experiments

Setups

Datasets The training of our models is carried out on the **Face Forensics++** (FF++) (Rössler et al. 2018) dataset. Here, we follow the official dataset division strategy. According to the compression ratios, FF++ can be divided into three subsets: RAW, HQ and LQ. Note that our experiments on FF++ are conducted on HQ and LQ only. To verify the effectiveness of our FADE comprehensively, FADE and the compared methods are tested not only on FF++, but also on **Celeb-DF** (CDF) (Li et al. 2020c) and **DeepFake Detection** (DFD) (Nick Dufour and Andrew Gully 2019).

Implementation Details 8 continuous frames with an interval of 8 are sampled from the videos as the input. For each video, four clips with the same number of frames are evenly sampled for training, validation and testing. MTCNN (Zhang et al. 2016) is employed to detect the faces and landmarks in each frame. Input images will be resized to 320×320 and the landmarks will be utilized to calculate the RoIs of the AUs.

In the training process, the batch size is set as 4 (clips). The initial learning rate of MDGM is 2e-4, while the learning rate of the backbone is one fifth of it. The learning rates will be divided by 2 when the AUC score plateaus for 3 epochs. The models are optimized via AdamW (Loshchilov and Hutter 2017) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 0.001$. These models are trained on four RTX 3080s for approximately 10000 iterations and the best model is selected according to the evaluations on the validation set.

Comparisons with the Existing Methods

To demonstrate the effectiveness of our method, a variety of methods are selected for comparison, including MesoNet (Afchar et al. 2018), Xception (Chollet 2017), EfficientNet (Tan and Le 2019), DSP-FWA (Li and Lyu 2019), SPSL (Liu et al. 2021), Face X-ray (Li et al. 2020a), Multi-Attention (Zhao et al. 2021), F3-Net (Qian et al. 2020), Two-Branch (Masi et al. 2020), UV-Transformer (Khan and Dai 2021), LipForensics(Haliassos et al. 2021), DIANet (Hu et al. 2021), PatchDiffusion (Zhang et al. 2022), RECCE (Cao et al. 2022), and Finfer (Hu et al. 2022). Note that Xception is employed as the backbone network in our FADE.

Intra-dataset Evaluation As shown in Tab. 1, we achieve the best performance on the LQ set in terms of both the video-level ACC and AUC metrics. Although our method gives a slightly worse result than UV-Transformer on the HQ set, which possesses much more model parameters than the proposed method and needs additional UV texture map inputs extracted from 3DMM, FADE still outperforms all the other methods. Compared to PatchDiffusion (Zhang et al. 2022), which also applies GNN to deepfake detection, our FADE selects the features of AUs as the input of MDGM, rather than the patches of feature maps. We believe that the differences in feature modeling contribute to the performance gap between PatchDiffusion and our FADE.

Cross-quality & Cross-dataset Evaluation Here, the models are trained on the FF++ HQ dataset, and the frame-level AUC score is employed as the metric. Note that the generalization ability against JPEG compression is assessed

Method	LQ		HQ	
	ACC	AUC	ACC	AUC
Xception*	86.42	89.30	95.97	97.87
EffNet*	86.21	88.30	95.73	97.51
ResNet*	83.43	86.59	95.30	98.12
F3-Net*	87.01	88.15	96.30	98.85
FADE(Xception)	92.89 (+6.47)	95.98 (+6.68)	98.33 (+2.36)	99.52 (+1.65)
FADE(EffNet)	91.33 (+5.12)	93.68 (+5.38)	97.50 (+1.77)	99.21 (+1.70)
FADE(ResNet)	91.21 (+7.78)	93.77 (+7.18)	97.55 (+2.25)	99.31 (+1.19)
FADE(F3-Net)	90.56 (+3.55)	93.37 (+5.22)	97.93 (+1.63)	99.37 (+0.52)

Table 3: Evaluations of integrating our work to the existing frame-level methods. The values in brackets are the performance gains. * denotes our implementations.

by testing the model on the highly compressed data. The DFD and Celeb-DF datasets are utilized to assess the generalization abilities against unseen manipulation methods. According to Tab. 2, DSP-FWA and Face X-ray, which are trained on the HQ data, suffer giant performance drops on the LQ data. Meanwhile, the performance of our FADE has only dropped to 83.33%, which is still decent. In the cross-dataset test, our method gives the best performance on DFD and outperforms most of the existing methods on Celeb-DF. It is worth noting that our FADE without the Content Branch has shown better generalization performance, which indicates that the Content Branch may lead FADE to overfitting to specific forgery methods.

Integration with Different Frame-level Methods We have integrated our work to several existing frame-level methods, including Xception (Chollet 2017), EfficientNet (Tan and Le 2019), ResNet (He et al. 2016), and F3-Net (Qian et al. 2020). The improvements are shown in Tab. 3, where we utilize the averaged value of the frame-level results as the video-level results for these frame-level methods. After integrating MDGM, the improvements of video-level AUC on the LQ set is 6.57%, 5.12%, 7.78%, and 5.22% for Xception, EfficientNet-b4, ResNet-50, and F3-Net, respectively, similar to the gains in ACC. On the HQ set, though the performances of the existing frame-level methods are relatively high, our method can still achieve an approximately 2% gain. These results demonstrate that our work can further boost the performances of the existing frame-level methods in video deepfake detection.

Ablation Study

In this subsection, we conduct a series of ablation studies on different components of FADE. Note that the following experiments are carried out on FF++ with Xception being the backbone.

Impacts of Action Units and ConvGCN To demonstrate the positive influences of our AUs and ConvGCN, Four models are constructed and compared in Tab. 4. According to the results, AUs can bring a 4.47% gain to AUC and a

AU	MDGM	ACC	AUC
✗	✗	86.42	89.30
✓	✗	89.14(+2.72)	93.77(+4.47)
✓	✓†	90.64(+4.22)	94.97(+5.67)
✓	✓	**92.89(+6.47)**	**95.98(+6.68)**

Table 4: Ablation evaluations on the FF++ LQ dataset. † indicates that the update operation in MDGM follows the original GCN.

Method	ACC	AUC
Xception	86.42	89.30
+Action Branch	87.82(+1.40)	92.72(+3.42)
+Action Branch+Mixer	90.57(+4.15)	94.54(+5.24)
+Content Branch	89.79(+3.37)	93.68(+4.38)
+MDGM	**92.89(+6.47)**	**95.98(+6.68)**
+MHSA	89.84(+3.42)	93.32(+4.02)
+MHSA †	87.37(+1.70)	89.99(+0.72)

Table 5: Ablation evaluations on the FF++ LQ dataset. † represents that the input of the MHSA block is the feature patches rather than features of the Action Units.

1.72% gain to ACC. When MDGM is further integrated, the model is further improved. Note that different update operations in MDGM tend to perform differently, i.e., the convolution-based update operation obviously outperforms the linear-based one, because our ConvGCN tends to capture more positional relations and maintain more spatial information than the standard GCN.

Impacts of Each Branch We separately switch off the Action Branch and the Content Branch to demonstrate their impacts on the FF++ LQ dataset. Since the widely used multi-head self-attention (MHSA) mechanism is considered as a variant of GNN (Joshi 2020), we also implement the self-attention block, and add two such blocks to the same backbone as a competitor. For fair comparison, all the models' training settings are identical. The number of heads in the MHSA models is set to 8, which equals to the number of graphs in MDGM. According to Tab. 5, the pattern-mixer in the Action Branch tends to offer more abundant action patterns by adjusting the weights on edges and combinations of the basic patterns. It can be observed that the incorporation of the proposed Action Branch and Content Branch are indeed useful. Compared to MHSA, our Content Branch achieves a similar result with fewer heads. The performance of MHSA drops with the feature patches, which may be induced by the redundant information in certain patches, such as the patches corresponding to the backgrounds.

Visualizations

Visualizations of Adjacency Matrices We visualize all the adjacency matrices to illustrate the mechanism of FADE. Fig. 4(a) shows a kind of dependencies that AUs prefer to aggregate features from AUs at the identical location in other frames, and there exists a significant difference along the temporal dimension. Figs. 4(b) and (c) also show certain dif-

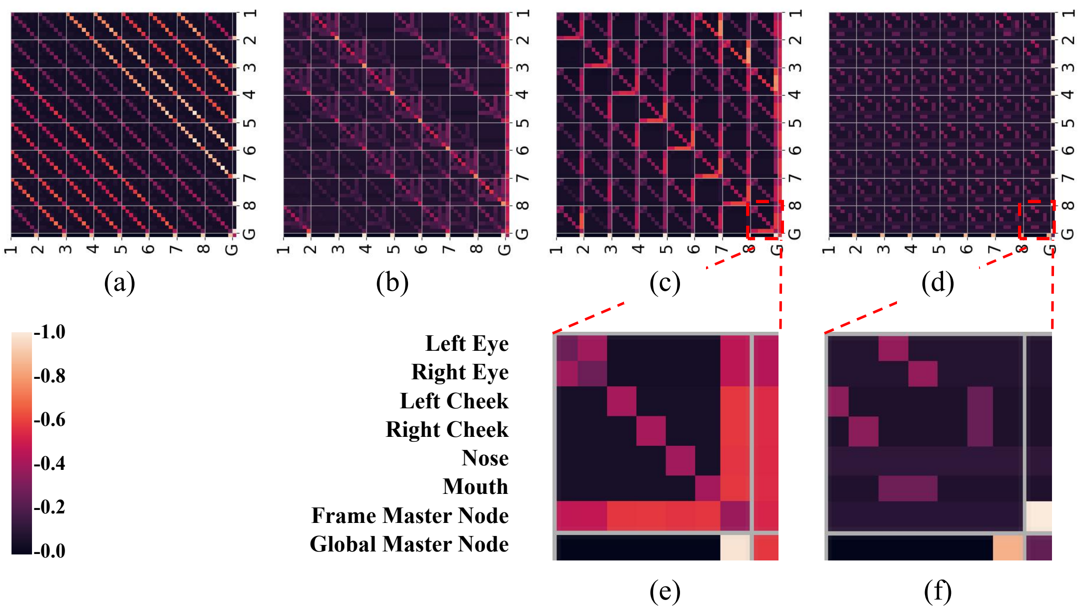

Figure 4: The visualizations of all the learned adjacency matrices in the Pattern Mixer. The horizontal and vertical axis coordinates represent the frame numbers, and G denotes the global master node.

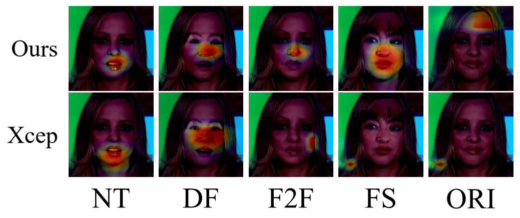

Figure 5: Several hard samples in the FF++ HQ dataset. The top row represents the heatmaps of our method, and the bottom row shows the heatmaps of Xception.

ferences in the temporal dimension, which are complementary to the dependencies shown in Fig. 4(a). In Fig. 4(d) the dependencies are almost evenly distributed along the temporal dimension. Details of the dependencies in a single frame are shown in Fig. 4(f). As can be observed, the left eye communicates with the left cheek and the left cheek communicates with the left eye and mouth, which is consistent with the positional relationships on a human face. In Fig. 4(e), the frame master node is evenly connected to all other AUs in the same frame, which significantly differs from other adjacency matrices.

Visualizations of Activations To better understand the decision-making mechanism of FADE, we provide the visualizations obtained via Grad-CAM (Selvaraju et al. 2017) in Fig. 5. When processing the real-like forgery data, the activations of Xception may not be in the facial regions. Compared to the baseline, since FADE focuses on the dependencies of facial action units, its heatmaps are mostly activated in the facial regions with frequent movements, even the forgery samples seem very authentic. In the same situation, the heatmaps of Xception are less reasonable. Fig. 6 shows the heatmaps against multiple forgery methods. It is

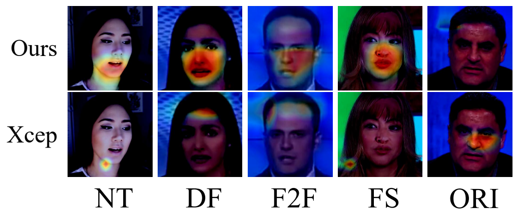

Figure 6: The activations vary along with the changes of forgery techniques. The upper row gives the visualizations of our FADE. The bottom row presents the heatmaps of Xception.

obvious that our FADE's activations are more compact and effective. However, since we discard the forehead regions, there may exists very slight activation near the forehead region in the benign frame. These results have explained the decision-making mechanism of our method and further proved the effectiveness of our FADE.

Conclusion

In this paper, we focus on exploiting the facial action dependencies among different facial regions in a video and formulate the deepfake video detection problem as a graph classification task. We propose Facial Action Dependency Estimation, to capture the subtle local clues and the intra- and inter-frame correlations of the facial action units. We propose Multi-Dependency Graph Module, which can capture abundant dependencies among the facial action units, and extracts clues in these dependencies, based on the correlations of action patterns and content of AUs. Extensive experiments have demonstrated the superiority and the interpretability of our FADE, and the potentials of integrating our work to the existing frame-level methods.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62272020, 61802391, 61972442, in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Natural Science Foundation of Tianjin of China under Grant 20JCYBJC00650, and in part by the Fundamental Research Funds for Central Universities.

References

Afchar, D.; Nozick, V.; Yamagishi, J.; and Echizen, I. 2018. Mesonet: a compact facial video forgery detection network. In *IEEE WIFS*, 1–7.

Cao, J.; Ma, C.; Yao, T.; Chen, S.; Ding, S.; and Yang, X. 2022. End-to-End Reconstruction-Classification Learning for Face Forgery Detection. In *IEEE CVPR*, 4113–4122.

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *IEEE CVPR*, 1251–1258.

Ciftci, U. A.; Demir, I.; and Yin, L. 2020a. FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals. *IEEE TPAMI*, 1–1.

Ciftci, U. A.; Demir, I.; and Yin, L. 2020b. How do the hearts of deep fakes beat? deep fake source detection via interpreting residuals with biological signals. In *IJCB*, 1–10.

Cozzolino, D.; Rössler, A.; Thies, J.; Nießner, M.; and Verdoliva, L. 2021. Id-reveal: Identity-aware deepfake video detection. In *IEEE ICCV*, 15108–15117.

Deepfakes. 2019. Deepfakes. https://github.com/deepfakes/faceswap. Accessed: 2019-09-18.

Dong, X.; Bao, J.; Chen, D.; Zhang, T.; Zhang, W.; Yu, N.; Chen, D.; Wen, F.; and Guo, B. 2022. Protecting Celebrities from DeepFake with Identity Consistency Transformer. In *IEEE CVPR*, 9468–9478.

Ekman, P.; and Friesen, W. V. 1978. Facial action coding system. *Environmental Psychology & Nonverbal Behavior*.

Fernandes, S.; Raj, S.; Ortiz, E.; Vintila, I.; Salter, M.; Urosevic, G.; and Jha, S. 2019. Predicting Heart Rate Variations of Deepfake Videos using Neural ODE. In *IEEE ICCVW*, 1721–1729.

Fridrich, J.; and Kodovsky, J. 2012. Rich models for steganalysis of digital images. *IEEE TIFS*, 7(3): 868–882.

Gu, Z.; Chen, Y.; Yao, T.; Ding, S.; Li, J.; Huang, F.; and Ma, L. 2021. Spatiotemporal inconsistency learning for deepfake video detection. In *ACMMM*, 3473–3481.

Haliassos, A.; Vougioukas, K.; Petridis, S.; and Pantic, M. 2021. Lips Don't Lie: A Generalisable and Robust Approach To Face Forgery Detection. In *IEEE CVPR*, 5039–5049.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *NIPS*, 30.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *IEEE ICCV*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE CVPR*, 770–778.

Hosler, B.; Salvi, D.; Murray, A.; Antonacci, F.; Bestagini, P.; Tubaro, S.; and Stamm, M. C. 2021. Do deepfakes feel emotions? A semantic approach to detecting deepfakes via emotional inconsistencies. In *IEEE CVPR*, 1013–1022.

Hu, J.; Liao, X.; Liang, J.; Zhou, W.; and Qin, Z. 2022. FInfer: Frame Inference-based Deepfake Detection for High-Visual-Quality Videos. *IEEE TPAMI*, 1–9.

Hu, Z.; Xie, H.; Wang, Y.; Li, J.; Wang, Z.; and Zhang, Y. 2021. Dynamic inconsistency-aware deepfake video detection. In *IJCAI*, 736–742.

Jain, A.; Zamir, A. R.; Savarese, S.; and Saxena, A. 2016. Structural-rnn: Deep learning on spatio-temporal graphs. In *IEEE CVPR*, 5308–5317.

Joshi, C. 2020. https://graphdeeplearning.github.io/post/transformers-are-gnns/. Accessed: 2020-02-12.

Khan, S. A.; and Dai, H. 2021. Video transformer for deepfake detection with incremental learning. In *ACMMM*, 1821–1828.

Li, L.; Bao, J.; Zhang, T.; Yang, H.; Chen, D.; Wen, F.; and Guo, B. 2020a. Face x-ray for more general face forgery detection. In *IEEE CVPR*, 5001–5010.

Li, X.; Lang, Y.; Chen, Y.; Mao, X.; He, Y.; Wang, S.; Xue, H.; and Lu, Q. 2020b. Sharp multiple instance learning for deepfake video detection. In *ACMMM*, 1864–1872.

Li, Y.; and Lyu, S. 2019. Exposing DeepFake Videos By Detecting Face Warping Artifacts. In *IEEE CVPRW*, 46–52.

Li, Y.; Yang, X.; Sun, P.; Qi, H.; and Lyu, S. 2020c. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *IEEE CVPR*, 3207–3216.

Liu, H.; Li, X.; Zhou, W.; Chen, Y.; He, Y.; Xue, H.; Zhang, W.; and Yu, N. 2021. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In *IEEE CVPR*, 772–781.

Liu, Z.; Qi, X.; and Torr, P. H. 2020. Global texture enhancement for fake face detection in the wild. In *IEEE CVPR*, 8060–8069.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lucey, P.; Cohn, J. F.; Kanade, T.; Saragih, J.; Ambadar, Z.; and Matthews, I. 2010. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *IEEE CVPRW*, 94–101.

Masi, I.; Killekar, A.; Mascarenhas, R. M.; Gurudatt, S. P.; and AbdAlmageed, W. 2020. Two-branch recurrent network for isolating deepfakes in videos. In *ECCV*, 667–684.

Mittal, T.; Bhattacharya, U.; Chandra, R.; Bera, A.; and Manocha, D. 2020. Emotions don't lie: An audio-visual deepfake detection method using affective cues. In *ACMMM*, 2823–2832.

Nick Dufour, G. R.; and Andrew Gully, J. 2019. Deepfakes Detection. https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html. Accessed: 2019-09.

Perov, I.; Gao, D.; Chervoniy, N.; Liu, K.; Marangonda, S.; Umé, C.; Dpfks, M.; Facenheim, C. S.; Luis, R.; Jiang, J.;

et al. 2020. DeepFaceLab: A simple, flexible and extensible face swapping framework. https://github.com/iperov/DeepFaceLab. Accessed: 2020-05.

Qi, H.; Guo, Q.; Juefei-Xu, F.; Xie, X.; Ma, L.; Feng, W.; Liu, Y.; and Zhao, J. 2020. Deeprhythm: Exposing deepfakes with attentional visual heartbeat rhythms. In *ACMMM*, 4318–4327.

Qian, Y.; Yin, G.; Sheng, L.; Chen, Z.; and Shao, J. 2020. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *ECCV*, 86–103.

Rössler, A.; Cozzolino, D.; Verdoliva, L.; Riess, C.; Thies, J.; and Nießner, M. 2018. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE ICCV*, 618–626.

Sun, Z.; Han, Y.; Hua, Z.; Ruan, N.; and Jia, W. 2021. Improving the efficiency and robustness of deepfakes detection through precise geometric features. In *IEEE CVPR*, 3609–3618.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 6105–6114.

Wale, N.; Watson, I. A.; and Karypis, G. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *KAIS*, 14(3): 347–375.

Wu, P.; Liu, J.; Shi, Y.; Sun, Y.; Shao, F.; Wu, Z.; and Yang, Z. 2020. Not only look, but also listen: Learning multimodal violence detection under weak supervision. In *ECCV*, 322–339.

Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*.

Zhang, B.; Li, S.; Feng, G.; Qian, Z.; and Zhang, X. 2022. Patch Diffusion: A General Module for Face Manipulation Detection. In *AAAI*, 3243–3251.

Zhang, K.; Zhang, Z.; Li, Z.; and Qiao, Y. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE SPL*, 23(10): 1499–1503.

Zhang, Y.; Wang, Y.; and Guo, Y. 2019. Scene Graph Generation via Convolutional Message Passing and Class-Aware Memory Embeddings. In *ICANN*, 620–633.

Zhang, Y.; Zheng, L.; and Thing, V. L. 2017. Automated face swapping and its detection. In *IEEE ICSIP*, 15–19.

Zhao, H.; Zhou, W.; Chen, D.; Wei, T.; Zhang, W.; and Yu, N. 2021. Multi-attentional deepfake detection. In *IEEE CVPR*, 2185–2194.

Zhong, J.-X.; Li, N.; Kong, W.; Liu, S.; Li, T. H.; and Li, G. 2019. Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In *IEEE CVPR*, 1237–1246.

Zi, B.; Chang, M.; Chen, J.; Ma, X.; and Jiang, Y.-G. 2020. Wilddeepfake: A challenging real-world dataset for deepfake detection. In *ACMMM*, 2382–2390.